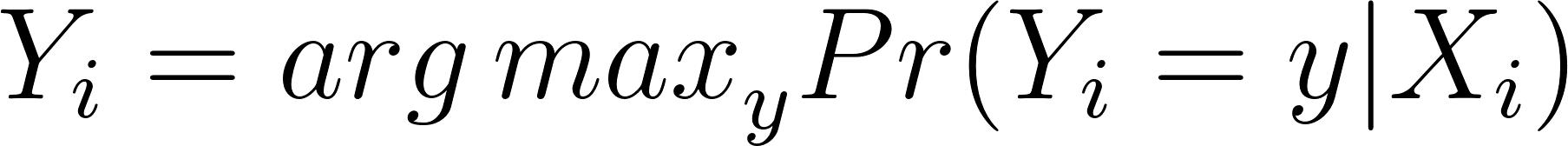
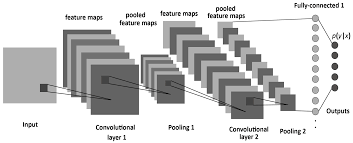
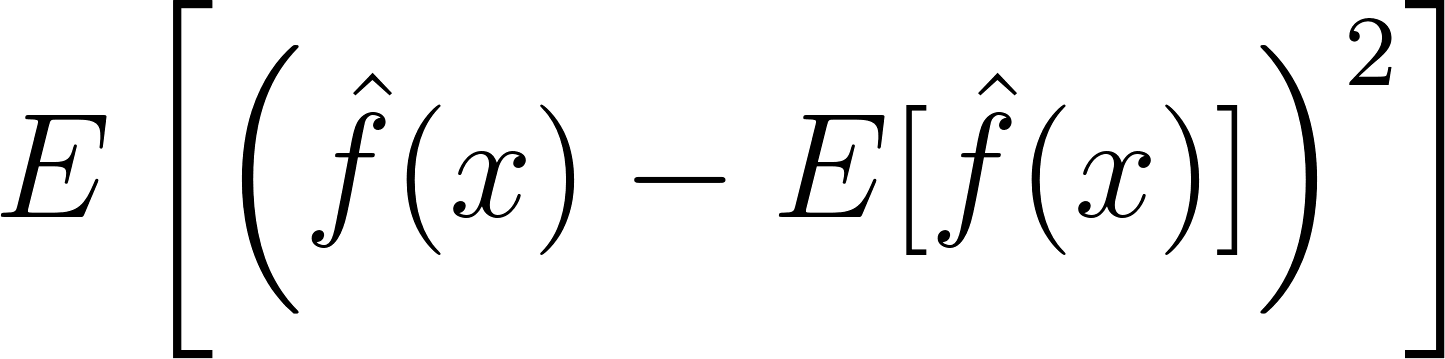
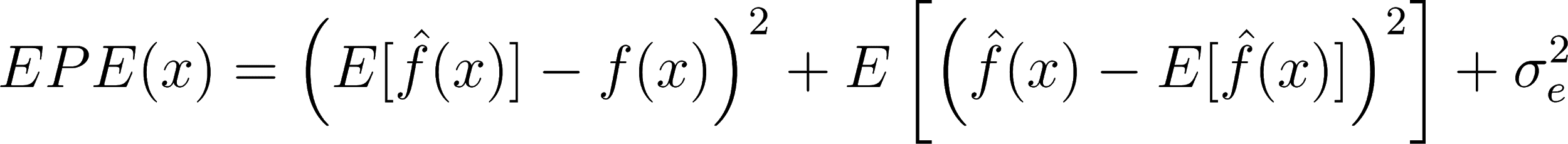
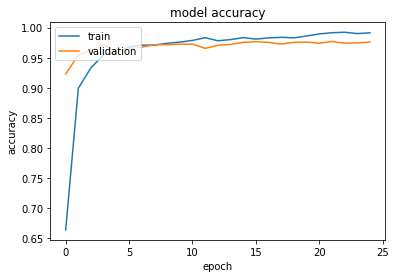
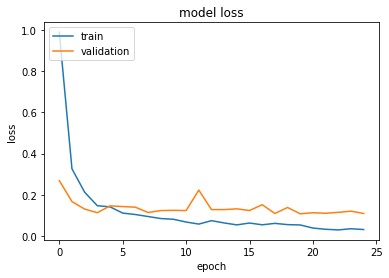
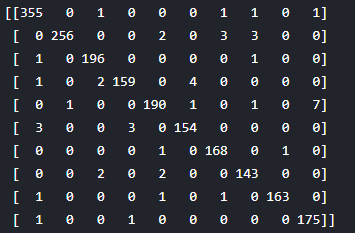
# Methodology:

* + Logistic Regression
    - Model the probability odds ratio of belong to certain class as linear combination of covariates, which gives rise to the following prediction function:
    - 
    - Since there are 10 different classes, we adopt one-versus-rest approach to make the classification. Specifically, for each sample we compute its probability of belonging to certain class and classify it to be the one that maximize the probability value.
    - [](about:blank)
    - After performing hyperparameter search, we figured out the optimal hyperparameter settings to be follows: inverse of regularization strength C as 0.3, penalty option as L2 and optimization algorithm to be limited memory Broyden–Fletcher–Goldfarb–Shanno (BFGS).
    - Hence the objective function of each class would be the followings:
    - 
  + Convolutional Neural Network
    - We utilize one of the most popular and effective model for image recognition which is Convolutional Neural Network(CNN) on MNIST dataset. The deep learning algorithms generally depend heavily on the architecture design, which can be conceptually described as following graph. In terms of CNN, its impressive performance in classifying images may be attributed to its feature extraction capability in convolutional layers, and then feature reconstruction capability in pooling layers. The trick of randomly dropping out specific input data during each iteration also enhance the possibility of removing noise of input images.
    - 
    - Careful experiments and tuning lead to our final architecture of the CNN model: 2 consecutive convolutional layers followed by 1 max-polling layer, and another 2 consecutive convolutional layers with 1 max-polling layer, and we also introduce artificial noise to input feature so as to create more data to feed model. Other hyperparameter setting may be referenced in our source code.

# Result & Analysis

|  |  |
| --- | --- |
| Model | Test accuracy |
| Logistic Regression | 0.91330 |
| CNN | 0.97608 |
|  |  |
|  |  |

* + - We trained the CNN model for 25 epochs and record the accuracy as well as its softmax loss value in following 2 graphs, where the validation refers to the result obtained on test data set. The confusion matrix for classification is also included, which shows that the model has difficulty in distinguishing between digit ‘3’ and ‘5’, and between ‘4’ and ‘9’.
    - As shown in the accuracy history, the accuracy of CNN model quickly converges above 0.95 within first 3 epochs, with a few fluctuation in the following epochs. One strange phenomenon is that training accuracy turns out to be lower than test accuracy in first 4 epochs. This might be explained by following reasonings: The adoption of noise data generation expands the training data set and also introduce some images that are more difficult to classify than the original one, thus lowering the performance of model on training data. However, such additional difficulties posed on training data actually effectively prevent the model from overfitting since the noise counteracts the original noise. This corresponds to reducing variance [](about:blank)of estimator in expected prediction error:
    - [](about:blank)
  + 
  + 
  + Confusion matrix
    - 

973094

0.973094

* Conclusion:
  + Based on the experiment results, we may conclude that CNN model is more suitable than the other 3 classifiers in terms of coping with image data. Such advantage is gained through its capability of extracting, abandoning and reconstructing hidden features through high dimensional image data. In contrast, the other 3 classifiers utilize all of the 256 pixels to compute the decision boundary, which is very prone to overfitting.